FAX COVER SHEET

TO	
COMPANY	
FAX NUMBER	15712702059
FROM	Lee & Hayes
DATE	2008-12-05 18:58:38 GMT
RE	10/657,463 | MS1-1596US | Proposed Agenda

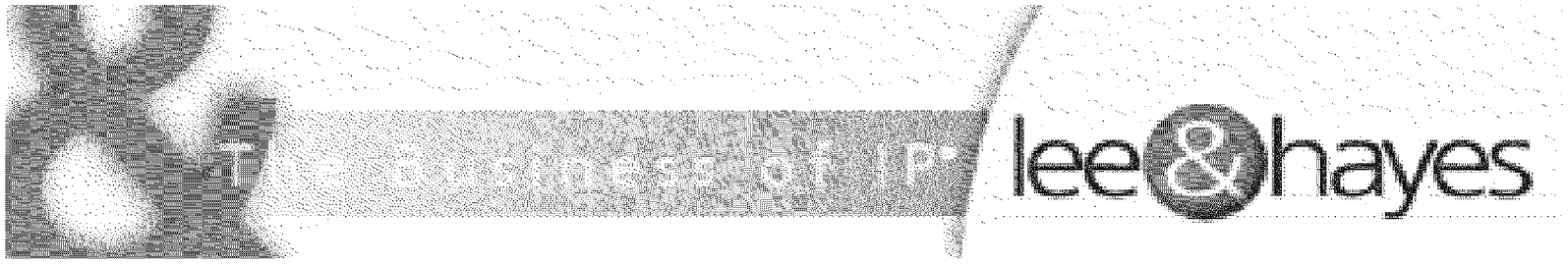
COVER MESSAGE

Cherri Simon
(509) 944-4776
cherri@leehayes.com<mailto:cherri@leehayes.com>

[cid:image001.jpg@01C956C8.605E4AD0]

Lee & Hayes pllc, Intellectual Property Law
601 West Riverside, Suite 1400, Spokane, WA 99201 | (509)323-8979 fax |
www.leehayes.com<http://www.leehayes.com/>

PAGE 1/23 * RCVD AT 12/5/2008 1:59:23 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-6/20 * DNIS:2702059 * CSID:15092672653 * DURATION (mm-ss):11-06


lee&hayes

INFORMAL COMMUNICATION: Please do not put in the file

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE (USPTO)

Serial Number	10/657,463
Confirmation Number	9824
Filing Date	Sep 8, 2003
Title of Application	Compiling Source Code Using Generic Classes
First Named Inventor	Makarand Gadre
Assignee	Microsoft Corporation
Group Art Unit	2192
Examiner	Zheng Wei
Attorney Docket Number	MS1-1596US
Nature of this Document	Informal Communication in Preparation for Scheduling an Examiner Interview

To: Examiner Wei
Fax: 571-270-2059
Phone: 571-270-1059

From: Ningning Xu
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201
ningning@leehayes.com
(Tel. 509-324-9256; Fax 509-323-8979)

Dear Examiner Wei:

[0001] This communication provides an agenda for an interview of this matter. The interview is currently scheduled for 2:30 pm Monday Dec. 8, 2008, in the Randolph building. We sincerely hope that it will not be necessary to reschedule the interview. However, in the unlikely event that it becomes necessary to reschedule, then please contact my assistant or me directly. Our contact info is on the signature page of this document. Thank you in advance for talking with me about this matter.

INFORMAL COMMUNICATION: Please do not put in the file

Interview Agenda:

- Double Patenting Rejection;
- Discussion of exemplary differences between the application/claims and the cited references; and
- Discussion of proposed amendments

Double Patenting

[0002] Claims 1, 13, and 18 were rejected on the ground of non-statutory obvious-type double patenting as being unpatentable over claims 1, 12, and 23 of co-pending Application No. 10/657,468. However, the co-pending application was abandoned as of November 24, 2008. Accordingly, withdrawal of the double patenting rejections to claims 1, 13, and 18 is respectfully requested.

Exemplary Differences

[0003] Claim 1, with proposed amendments, recites (in part with emphasis added):

> ***the plurality of types comprise an integer type, a float type, and a string type***;

[0004] In contrast, the cited reference (the article written by Eric Allen) in page 3 indicated that:

> One limitation to type variables in Tiger is that they must be instantiated with reference types – primitive types won't work. ...What we'd really like to see would be automatic boxing and unboxing of primitive types, similar to what is done in C# (except

better). Unfortunately, Tiger is not scheduled to include autoboxing of primitives (but one can always hope for Java 1.6!).

[0005] Eric in Section "Limitations on generic types" at page 8 of the cited reference further provides:

> Let's start by reviewing the limitations on the use of generic types in Tiger and JSR-14: ...
>
> ***Generic type parameters can't be instantiated with primitive types***...

[0006] It is known in the art that types including an integer type, a float type, and a string type are primitive types in Java language. These types are supported in the generic class provided in the Application including, not limited to, original claims 30 and 48, and table 2 at pages 19 of the Specification.

[0007] I would like to discuss claims 18, 31, and 40 in view of proposed amendments to claim 1.

[0008] Independent claim 15, with proposed amendments, recites (in part with emphasis added):

> ***the one of the plurality of second classes is a generic class nested within the first class***

[0009] The article written by Eric Allen is silent with respect to nesting a generic class within another generic class. Support for nesting generic class in the instant matter can be found in original claims, including claims 23 and 24, and portion of the specification at page 20, lines 9-17.

INFORMAL COMMUNICATION: Please do not put in the file

[0010] I would also like to discuss claims25, 36, and 51 in view of proposed amendments to claim 15.

Proposed Amendments

[0011] Please see the attached Appendix of Proposed Claim Amendments. I would like to discuss your opinion regarding the proposed amendments in light of the currently cited references.

[0012] Thank you in advance for scheduling time for this interview. I look forward to discussing this with you.

Respectfully Submitted,

Dated: December 5, 2008

By:_____

Ningning Xu
Reg. No. L0293
(509) 944-4726
ningning@leehayes.com
www.leehayes.com

Beatrice L. Koempel-Thomas
Reg. No. 58213
509-944-4759
bea@leehayes.com

Assistant: Cherri Simon
(509) 944-4776
cherri@leehayes.com

INFORMAL COMMUNICATION: Please do not put in the file

Appendix of Claims with Proposed Amendments

1. (Proposed amended) A method of generating common intermediate language code for use in a framework, the method comprising:

receiving a portion of JAVA™ language source code referencing, through a generic class syntax, one or more generic classes unspecified in a formal JAVA™ language specification, wherein:

each of the one or more generic classes refers to a first class configured to operate uniformly on ~~values~~ instances of a plurality of ~~different~~ types associated with the first class ~~and defined by a plurality of second classes~~;

the plurality of types comprise an integer type, a float type, and a string type; and

the generic class syntax is not specified in the formal JAVA™ language specification and identifies one instance of the plurality of types ~~second classes~~ by surrounding the one instance ~~of the plurality of second classes~~ with angular brackets following the first class; and

generating, through a first compiler different from a formal compiler complying with the formal JAVA™ language specification, language-neutral intermediate language code representing the portion of JAVA™ language source code referencing the one or more generic classes.

INFORMAL COMMUNICATION: Please do not put in the file

2. (Previously Presented) A method as recited in claim 1 further comprising parsing the portion of the JAVA™ language source code into a parse tree representing the portion of the JAVA™ language source code before compiling the portion with the first class.

3. (Proposed amended) A method as recited in claim 2 further comprising nesting a constructed class of the first class in the parse tree, wherein the constructed class is a generic class nested within the first class.

4. (Proposed amended) A method as recited in claim 1 further comprising:

generating a parse tree having a token referencing the first class and a token referencing the one instance ~~one of the plurality of second classes~~; and

semantically analyzing the parse tree to determine validity of semantics of the first class.

5. (Original) A method as recited in claim 4 wherein the semantically analyzing comprises determining whether operations applied to the first class are valid.

6. (Original) A method as recited in claim 1 further comprising generating metadata descriptive of the first class.

INFORMAL COMMUNICATION: Please do not put in the file

7. (Previously Presented) A method as recited in claim 6 further comprising storing the metadata with the language-neutral intermediate language code, whereby the language-neutral intermediate language code is used by an application program.

8. (Proposed amended) A method as recited in claim 1 further comprising creating a compiled project including the language-neutral intermediate language code and metadata descriptive of the first class and the one instance ~~of the plurality of second classes~~.

9. (Original) A method as recited in claim 1 further comprising executing the language-neutral intermediate language code with a runtime engine.

10. (Canceled).

11. (Previously Presented) A method as recited in claim 1 wherein the framework is a .NET™ Framework.

INFORMAL COMMUNICATION: Please do not put in the file

12. (Previously Presented) A method as recited in claim 11 wherein the developing comprises authoring the portion of JAVA™ language source code with a VISUAL J# .NET™ application of the .NET™ Framework.

13. (Proposed amended) A method of compiling in a framework, the method comprising:

receiving a portion of JAVA™ language software having a declaration of an instance of a generic class unspecified in a formal JAVA™ language specification, the declaration of the instance of the generic class being implemented through a generic class syntax, wherein:

the generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

the generic class syntax is not specified in the formal JAVA™ language specification and identifies one of the plurality of second classes by surrounding the one of the plurality of second classes with angular brackets following the first class; and

the one of the plurality of second classes is a generic class nested within the first class; and

parsing the declaration into a token corresponding to the generic class; and

creating an intermediate language code block corresponding to the parsed declaration through a first compiler other than a traditional compiler complying with the formal JAVA™ language specification, wherein:

the intermediate language code block containing the instance of the generic classes is made executable by a runtime engine.

14. (Original) A method as recited in claim 13 further comprising associating the declaration of the instance of the generic class with a defined generic class in a generic class library.

15. (Original) A method as recited in claim 14 further comprising tokenizing a parse tree with an identifier corresponding to the defined generic class, the parse tree comprising a hierarchical representation of the declaration.

16. (Original) A method as recited in claim 13 further comprising creating metadata describing the portion of the JAVA™ language software.

17. (Original) A method as recited in claim 14 further comprising validating an operation on the instance of the generic class based on the defined generic class.

INFORMAL COMMUNICATION: Please do not put in the file

18. (Proposed amended) A computer-readable medium having stored thereon computer-executable instructions for performing a method of compiling in a framework, the method comprising:

receiving a portion of JAVA™ language software including an instruction that references a generic class of a specified type through use of a generic class syntax, wherein:

the generic class is unspecified in a formal JAVA™ language specification and refers to a first class configured to operate uniformly on instances of a plurality of ~~values of different~~ types associated with the first class ~~and defined by a plurality of second classes~~;

the plurality of types comprise an integer type, a float type, and a string type; and

the generic class syntax is not specified in the formal JAVA™ language specification and identifies one of the instances of the plurality of types ~~second classes~~ by surrounding the one instance ~~of the plurality of second classes~~ with angular brackets following the first class; and

creating a parse tree having a generic class identifier associated with the generic class and type identifier associated with the specified type; and

generating, through a first compiler other than a traditional compiler complying with the formal JAVA™ language specification, one or more intermediate language instructions representing the JAVA™ language instruction based on the parse tree .

INFORMAL COMMUNICATION: Please do not put in the file

19. (Proposed amended) A computer-readable medium as recited in claim 18, wherein the method further comprises ~~comprising~~ translating the one or more intermediate language instructions into microprocessor-specific binary for execution by a computer.

20. (Proposed amended) A computer-readable medium as recited in claim 18, wherein the method further comprises ~~comprising~~ validating the parse tree according to a generic class definition associated with the generic class.

21. (Original) A computer-readable medium as recited in claim 20, wherein validating the parse tree comprises determining whether an assignment applied to the instance of the generic class assigns an allowable type to the instance.

22. (Proposed amended) A computer-readable medium as recited in claim 18, wherein the method further comprises ~~comprising~~ generating metadata associated with the generic class.

23. (Original) A computer- readable medium as recited in claim 18, wherein the specified type is a second generic class of a second specified type.

INFORMAL COMMUNICATION: Please do not put in the file

24. (Original) A computer- readable medium as recited in claim 23, wherein the method further comprises nesting the second generic class and the second specified type at different levels in a hierarchy in the parse tree.

25. (Proposed amended) A computer-readable medium having stored thereon intermediate language code block executable by a runtime engine, the intermediate language code block generated from a representation of a portion of source code for use by a compiler in a framework, the representation comprising:

a generic class identifier field having data identifying, through use of a generic class syntax, a generic class referenced in the portion of source code in a first programming language for which the generic class syntax is not formally specified, wherein:

the referenced generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes; and

the generic class syntax identifies one of the plurality of second classes by surrounding the one of the plurality of second classes with angular brackets following the first class; and

a constructed class identifier field having data identifying a constructed class of the generic class, wherein the constructed class of the generic class nests another generic class within the generic class.

INFORMAL COMMUNICATION: Please do not put in the file

26. (Proposed cancelled) ~~A computer-readable medium as recited in claim 25, wherein the representation further comprises:~~

~~at least one nested constructed class that is a generic class~~.

27. (Original) A computer-readable medium as recited in claim 25, wherein the generic class identifier identifies a Queue class.

28. (Previously Presented) A computer-readable medium as recited in claim 25, wherein the first programming language is a JAVA™ language.

29. (Previously Presented) A computer-readable medium as recited in claim 25, wherein the representation further comprises metadata describing the generic class.

30. (Original) A computer-readable medium as recited in claim 25, wherein the constructed class comprises one of:

an integer type;

a float type;

a Stack type;

a Queue type; and

a Dictionary type.

INFORMAL COMMUNICATION: Please do not put in the file

31. (Proposed amended) A method of compiling in a framework, the method comprising:

receiving a portion of source code in a first programming language for which one or more generic types are not specified in a formal definition of the first programming language, wherein:

each of the one or more generic types refers to a first type configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second types;

the plurality of second types comprise an integer type, a float type, and a string type; and

each of the one or more generic types uses a generic type syntax not specified in the formal definition of the first programming language;

parsing the portion of source code into a parse tree comprising each instance of the one or more generic types in the portion of source code, wherein each instance of the one or more generic types comprises:

the first type; and

at least one instance of one of the plurality of second types associated with the first type; and

generating an intermediate representation of the parse tree representing the parse tree .

INFORMAL COMMUNICATION: Please do not put in the file

32. (Previously Presented) The method as recited in claim 31 further comprising importing metadata describing the first type and the at least one instance of one of the plurality of second types associated with the first type.

33. (Previously Presented) The method as recited in claim 31 further comprising tokenizing the parse tree with a token corresponding to the one or more generic types .

34. (Previously Presented) The method as recited in claim 33 further comprising tokenizing the parse tree with at least one token corresponding to the at least on instance of one of the plurality of second types associated with the first type .

35. (Previously Presented) The method as recited in claim 31 wherein each of the one or more generic types is a .NET™ generic class.

36. (Proposed amended) A method of generating microprocessor-executable code in a framework, the method comprising:

receiving a portion of source code written in a first programming language for which generic classes are unspecified, the portion of source code including a generic class declaration declaring a generic class, wherein:

the generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

the generic class uses a generic class syntax not specified in a formal specification of the first programming language;

the generic class declaration creates a constructed class of the generic class by associating a reference of one of the plurality of second classes with the generic class; and

the one of the plurality of second classes is a another generic class nested within the generic class; and

generating a module having microprocessor-executable instructions corresponding to the constructed class based on the portion of source code , the module further having metadata describing the constructed class.

37. (Original) A method as recited in claim 36 wherein the microprocessor-executable instructions comprise intermediate language instructions.

38. (Original) A method as recited in claim 36 wherein the microprocessor-executable instructions comprise Microsoft® Intermediate Language instructions.

INFORMAL COMMUNICATION: Please do not put in the file

39. (Original) A method as recited in claim 36 wherein the metadata comprises at least one of:

a name of the constructed class;

visibility information indicating the visibility of the constructed class;

inheritance information indicating a class from which the constructed class derives;

interface information indicating one or more interfaces implemented by the constructed class;

method information indicating one or more methods implemented by the constructed class;

properties information indicating identifying at least one property exposed by the constructed class; and

events information indicating at least one event the constructed class provides.

40. (Proposed amended) A method of compiling in a framework, the method comprising:

receiving a portion of source code written in a first programming language for which one or more generic classes are unspecified in a formal language specification of the first programming language, wherein:

each of the one or more generic classes refers to a first class configured to operate uniformly on instances of a plurality of ~~values of different~~ types associated with the first class ~~and defined by a plurality of second classes~~;

the plurality of types comprise an integer type, a float type, and a string type; and

each of the one or more generic classes uses a generic class syntax unspecified in the formal language specification of the first programming language; and

generating an intermediate language representation of the portion of the source code in the first programming language through a first compiler other than a traditional compiler complying with the formal language specification of the first programming language~~, the intermediate representation having an instance of the first class and an instance of the at least one of the plurality of second classes~~.

41. (Canceled).

42. (Previously Presented) A method as recited in claim 40 wherein the first programming language is a JAVA™ language.

43. (Original) A method as recited in claim 40 further comprising validating the type based on a definition of the first class.

INFORMAL COMMUNICATION: Please do not put in the file

44. (Original) A method as recited in claim 43 further comprising validating an operation on the first class based on a definition of the first class.

45. (Original) A method as recited in claim 40 further comprising interpreting the intermediate representation for execution by a microprocessor.

46. (Proposed amended) A method as recited in claim 40 wherein angular brackets surround the at least one of the instances of the plurality of types ~~second classes~~ associated with a reference of the first class.

47. (Original) A method as recited in claim 40 wherein the first class is a Queue class.

48. (Proposed amended) A method as recited in claim 47 wherein the ~~at least one of~~ the plurality of types further comprise ~~second classes comprises one of~~:

~~an int type;~~

~~a string type; and~~

a Queue type.

INFORMAL COMMUNICATION: Please do not put in the file

49. (Original) A method as recited in claim 48 wherein the Queue type includes at least one nested class reference referencing a second type associated with the Queue type.

50. (Proposed amended) A method as recited in claim 40 wherein:

the plurality of types further comprise ~~at least one of the plurality of~~ second class; and

the second class ~~classes associated with a reference of the first class~~ includes one or more nested generic class references.

51. (Proposed amended) A system for compiling in a framework, the system comprising:

a parser receiving JAVA™ language source code having an instruction referencing a generic class in a generic class syntax and specifying a type of the generic class, the parser further creating a parse tree from the JAVA™ language source code, the parse tree including a first node representing the generic class and a second node representing the specified type of the generic class, wherein :

the generic class refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes;

INFORMAL COMMUNICATION: Please do not put in the file

the generic class syntax is unspecified in the formal language specification of JAVA™ programming language and supported in the framework; and

each of the plurality of second classes is another generic class nested within the generic class; and

a code generator generating intermediate language code representing the JAVA™ language source code referencing the generic classes .

52. (Original) A system as recited in claim 51 further comprising:

a common intermediate language importer providing tokens associated with the generic class and the specified type of the generic class.

53. (Original) A system as recited in claim 51 further comprising a runtime engine executing the intermediate language code.

54. (Original) A system as recited in claim 51 further comprising a semantic analyzer analyzing the specified type to determine whether the specified type is an allowable type of the generic class.